IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A method for developing a real-time operating system, comprising:

specifying a set of n tasks, task(1) through task(n), to be scheduled for execution;

specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

using a data processor to synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks~~, said synthesized source code being executable on a target system after compilation~~; and

synthesizing source code from commands embedded in source code to control execution of said set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code, the synthesized source code implementing the real-time operating system and being executable on a target system after compilation.

2. (Cancelled)

3. (Previously Presented) The method of claim 1) including specifying f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

4. (Previously Presented) The method of claim 1) including specifying p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

5. (Previously Presented) The method of claim 1) including specifying c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

6. (Previously Presented) The method of claim 1) including specifying r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

7. (Previously Presented) The method of claim 1) where tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

Claims 8-14 (Canceled).

15. (Currently Amended) An apparatus for developing a real-time operating system comprising:

a computer;

a non-transitory computer readable medium in data communication with the computer, the computer readable medium including a software synthesis program stored thereon, which when executed by the computer causes the computer to specify a set of n tasks, task(1) through task(n), to be scheduled for execution; specify t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n; synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks~~, said synthesized source code being executable on a target system after compilation~~; and synthesize source code from commands embedded in source code to control

execution of said set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code, the synthesized source code implementing the real-time operating system and being executable on a target system after compilation.

16. (Cancelled)

17. (Previously Presented) The apparatus of claim 15 being configured to specify f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

18. (Previously Presented) The apparatus of claim 15 being configured to specify p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

19. (Previously Presented) The apparatus of claim 15 being configured to specify c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

20. (Previously Presented) The apparatus of claim 15 being configured to specify r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the preemptive-task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

21. (Previously Presented) The apparatus of claim 15 wherein tasks are given priority values

such that whenever the task scheduler chooses between scheduling multiple tasks, all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

22. (Currently Amended) An apparatus for developing a real-time operating system comprising:

a computer;

a non-transitory computer readable medium in data communication with the computer, the computer readable medium including a software synthesis program stored thereon, the software synthesis program including:

means for specifying a set of n tasks, task(1) through task(n), to be scheduled for execution;

means for specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

means for synthesizing source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks~~, said synthesized source code being executable on a target system after compilation~~; and

means for synthesizing source code from commands embedded in source code to control execution of said set of t init-tasks, wherein the means for synthesizing source code from commands embedded in source code includes means for generating new source code based on the commands embedded in source code, the synthesized source code implementing the real-time operating system and being executable on a target system after compilation.

23. (Cancelled)

24. (Previously Presented) The apparatus of claim 22 including means for specifying f f-loop tasks, each having have an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

25. (Previously Presented) The apparatus of claim 22 including means for specifying p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

26. (Previously Presented) The apparatus of claim 22 including means for specifying c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

27. (Previously Presented) The apparatus of claim 22 including means for specifying r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the preemptive-task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

28. (Previously Presented) The apparatus of claim 22 wherein tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which are ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.

29. (Currently Amended) A non-transitory machine-readable medium embodying instructions which, when executed by a machine, cause the machine to:

specify a set of n tasks, task(1) through task(n), to be scheduled for execution;

specify t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

synthesize source code from commands embedded in source code to implement the task scheduler for controlling execution of said set of n tasks, the task scheduler further

controlling one execution of each of said set of t init-tasks~~, said synthesized source code being executable on a target system after compilation~~; and

synthesize source code from commands embedded in source code to control execution of said set of t init-tasks, wherein synthesizing source code from commands embedded in source code includes generating new source code based on the commands embedded in source code, the synthesized source code implementing the real-time operating system and being executable on a target system after compilation.

30. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify f f-loop tasks, each having an associated integer value c(i) for i ranging from 1 to f and f being less than or equal to n, said task scheduler including a continuously executing loop such that each f-loop task executes exactly once every c(i) times that the loop is executed.

31. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify p p-loop tasks, each having an associated integer value t(i) for i ranging from 1 to p and p being less than or equal to n, the number t(i) representing a number of regular time units, said task scheduler including a timer that schedules each p-loop task i to be executed approximately once every t(i) time units.

32. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify c call-tasks, c being less than or equal to n, said task scheduler scheduling a call-task when another task requests that said call-task be executed.

33. (Previously Presented) The machine-readable medium of claim 29 being further configured to specify r preemptive-tasks, r being less than or equal to n, said task scheduler including a timer mechanism that counts a specified period of time at which time if a preemptive-task is currently executing, the task's state is stored and execution is given to said task scheduler to schedule another task until a later time when the task scheduler restores the state of said preemptive-task and execution of said preemptive-task is continued.

34. (Previously Presented) The machine-readable medium of claim 29 wherein tasks are given priority values such that whenever the task scheduler chooses between scheduling multiple tasks, all of which being ready to be executed, said task scheduler chooses from among those tasks that have the highest priority values.